

Data Reporting Techniques with Microsoft SQL Server

With Solutions

RMAIR Workshop
October, 2011
Albuquerque, NM

Mike Ellison
mike.ellison@unlv.edu

Contents

Brief Tour of SQL Server Tools	3
Task 1 - Create Listing Reports	4
Task 2 - Create Headcount and Other Aggregation Reports	6
Task 3 - Create Distribution Reports	9
Time Permitting - Create Parameterized Reports	12
Time Permitting – Modify Data	13

Brief Tour of SQL Server Tools

Objectives

- Introduce SQL Server Management Studio
- Browse the objects in a relational database
- Use a Query Window to issue database commands

Task 1 - Create Listing Reports

Objectives

- Use the SELECT command to display detail records
- Use the WHERE clause to apply criteria
- Use the ORDER BY clause to sort results
- Use the FROM clause and JOIN statements to display data from multiple tables
- Use table aliases

Report #1 – List all departments.

DepartmentCode	DepartmentDesc	MailStop	Phone
ACC	Accounting	4423	555-0394
BIO	Biology	2034	555-0001
CHE	Chemistry	4122	555-4324
EGR	Engineering	3094	555-3459
ENG	English	1234	555-1234
MAT	Math	4032	555-2345
MUS	Music	1109	555-9375
OUCH	Percussive Maintenance	1022	555-6824

```
SELECT *  
FROM Departments
```

Report #2 – List students in the English department sorted alphabetically.

LastName	FirstName	Department
Abercrombie	Euan	ENG
Abrams	Jessica	ENG
Anderson	Mitch	ENG
Applewhite	Betty	ENG
Araz	Dina	ENG
Atherton	Tish	ENG
Auda	Yusuf	ENG
Ayres	Colin	ENG
Bagnold	Millicent	ENG
Bailey	Will	ENG
...

```
SELECT LastName, FirstName, Department  
FROM Students  
WHERE Department = 'ENG'  
ORDER BY LastName, FirstName
```

Report #3 – List Freshmen older than 22 and their department codes.

LastName	FirstName	Age	Standing	Department
Abbott	Hannah	43	U01	EGR
Baddock	Malcolm	38	U01	BIO
Bell	Katie	53	U01	MAT
Borage	Libatius	44	U01	CHE
Dobbs	Emma	25	U01	ENG
Higgs	Bertie	50	U01	MAT
Potter	Lily	43	U01	MUS
Rookwood	Augustus	54	U01	BIO
Widdershins	Willy	31	U01	MAT
Hoynes	John	37	U01	MUS
...

```
SELECT LastName, FirstName, Age, Standing
       Department
FROM Students
WHERE Age > 22
      AND Standing = 'U01'
```

Report #4 – Modify the previous report to include the full department name rather than the department code.

LastName	FirstName	Age	Standing	DepartmentDesc
Abbott	Hannah	43	U01	Engineering
Baddock	Malcolm	38	U01	Biology
Bell	Katie	53	U01	Math
Borage	Libatius	44	U01	Chemistry
Dobbs	Emma	25	U01	English
Higgs	Bertie	50	U01	Math
Potter	Lily	43	U01	Music
Rookwood	Augustus	54	U01	Biology
Widdershins	Willy	31	U01	Math
Hoynes	John	37	U01	Music
...

```
SELECT s.LastName, s.FirstName, s.Age, s.Standing
       ,d.DepartmentDesc
FROM Students s
     INNER JOIN Departments d
     ON s.Department = d.DepartmentCode
WHERE s.Age > 22
      AND s.Standing = 'U01'
```

Task 2 - Create Headcount and Other Aggregation Reports

Objectives

- Use common aggregation functions to summarize multiple rows of data
- Use the AS keyword to supply alias column headings
- Use the GROUP BY clause to summarize data across groups

Report #1 – Produce a “Total Headcount” report.

TotalHeadcount
677

```
SELECT Count(*) as TotalHeadcount
FROM Students
```

Report #2 – Produce a “Headcount by Department” report.

Department	Headcount
NULL	2
ACC	55
EGR	81
CHE	85
ENG	154
MAT	158
BIO	70
MUS	72

```
SELECT Department, Count(*) as Headcount
FROM Students
GROUP BY Department
```

Report #3 – Add a subgrouping for a “Headcount by Department and Ethnicity” report, with sorting.

Department	Ethnicity	Headcount
ACC	A	5
ACC	B	4
ACC	C	31
ACC	D	9
ACC	E	3
ACC	F	3
BIO	A	2
BIO	B	10
BIO	C	33
BIO	D	14
...

```
SELECT Department, Ethnicity
       , Count(*) as Headcount
FROM Students
GROUP BY Department, Ethnicity
ORDER BY Department, Ethnicity
```

Report #4 – Determine the average age of all students as well as the youngest and oldest.

Average	Youngest	Oldest
23.05	4	55

```
SELECT Avg(Age) AS Average
       , Min(Age) AS Youngest
       , Max(Age) AS Oldest
FROM Students
```

Report #5 – Display the total student credit hours by department.

Department	SCH
NULL	6
ACC	627
BIO	790
CHE	967
EGR	955
ENG	1775
MAT	1909
MUS	862

```
SELECT Department
       , Sum(Credits) as SCH
FROM Students
GROUP BY Department
```

How could Report 5 be modified to use the full department names rather than the codes?

Department	SCH
Accounting	627
Biology	790
Chemistry	967
Engineering	955
English	1775
Math	1909
Music	862

```
SELECT d.DepartmentDesc
       , Sum(s.Credits) as SCH
FROM Students s INNER JOIN Departments d
       On s.Department = d.DepartmentCode
GROUP BY d.DepartmentDesc
```

Task 3 - Create Distribution Reports

Objectives

- Use the CASE...END statement to create a distribution, or “bucket” column
- Use the WHEN, THEN, and ELSE keywords
- Use a subquery
- Create a View

Report #1 – Produce a query that categorizes students based on age.

StudentID	FirstName	LastName	Age	AgeCategory
S00001	Hannah	Abbott	43	36-45
S00002	Euan	Abercrombie	20	20-22
S00003	Stewart	Ackerley	22	20-22
S00004	Bertram	Aubrey	52	46 and older
S00005	Malcolm	Baddock	38	36-45
S00006	Ludo	Bagman	21	20-22
S00007	Otto	Bagman	20	20-22
S00008	Millicent	Bagnold	18	17-19
S00009	Bathilda	Bagshot	20	20-22
S00010	Ali	Bashir	45	36-45
...

```
SELECT StudentID, FirstName, LastName, Age
, CASE
    WHEN Age >= 17 AND Age <= 19 THEN '17-19'
    WHEN Age >= 20 AND Age <= 22 THEN '20-22'
    WHEN Age >= 23 AND Age <= 26 THEN '23-26'
    WHEN Age >= 27 AND Age <= 35 THEN '27-35'
    WHEN Age >= 36 AND Age <= 45 THEN '36-45'
    WHEN Age >= 46 THEN '46 and older'
    ELSE 'Younger than 17'
END AS AgeCategory
FROM Students
```

Report #2 – Using query #1 as a subquery, produce a “Distribution by Age” report.

AgeCategory	Headcount
17-19	200
20-22	292
23-26	101
27-35	21
36-45	33
46 and older	28

```
SELECT AgeCategory, Count(StudentID) as Headcount
FROM
(
    SELECT StudentID, FirstName, LastName, Age
    ,CASE
        WHEN Age >= 17 AND Age <= 19 THEN '17-19'
        WHEN Age >= 20 AND Age <= 22 THEN '20-22'
        WHEN Age >= 23 AND Age <= 26 THEN '23-26'
        WHEN Age >= 27 AND Age <= 35 THEN '27-35'
        WHEN Age >= 36 AND Age <= 45 THEN '36-45'
        WHEN Age >= 46 THEN '46 and older'
        ELSE 'Younger than 17'
    END AS AgeCategory
    FROM Students
) Ranges
GROUP BY AgeCategory
ORDER BY AgeCategory
```

Report #3 – To simplify execution of this Age Distribution report, create a View that encapsulates the query.

```
CREATE VIEW DistributionByAge AS
SELECT AgeCategory, Count(StudentID) as Headcount
FROM
(
    SELECT StudentID, FirstName, LastName, Age
        ,CASE
            WHEN Age >= 17 AND Age <= 19 THEN '17-19'
            WHEN Age >= 20 AND Age <= 22 THEN '20-22'
            WHEN Age >= 23 AND Age <= 26 THEN '23-26'
            WHEN Age >= 27 AND Age <= 35 THEN '27-35'
            WHEN Age >= 36 AND Age <= 45 THEN '36-45'
            WHEN Age >= 46 THEN '46 and older'
            ELSE 'Younger than 17'
        END AS AgeCategory
    FROM Students
) Ranges
GROUP BY AgeCategory
```

```
SELECT * FROM DistributionByAge
```

Time Permitting - Create Parameterized Reports

Objectives

- Create scripts that define parameters with the DECLARE and SET keywords
- Create stored procedures

Report #1 – Create a parameterized version of the “Student Listing” report, allowing a department code as input. Optionally, retrieve information about the given Department as well.

```
declare @dept varchar(4)
set @dept = 'MAT'

SELECT *
    FROM Departments
    WHERE DepartmentCode = @dept

SELECT *
    FROM Students
    WHERE Department = @dept
```

Report #2 – Use the script in #1 as a basis for a stored procedure. Create and execute the procedure.

```
CREATE PROCEDURE DepartmentListing
(
    @dept varchar(4)
)
AS
BEGIN
    SELECT *
        FROM Departments
        WHERE DepartmentCode = @dept

    SELECT *
        FROM Students
        WHERE Department = @dept
END
```

```
EXECUTE DepartmentListing 'ENG'
```

Time Permitting – Modify Data

Objectives

- Use a CREATE TABLE statement to build a new table to hold data
- Use the INSERT INTO statement to add records to the table
- Use the UPDATE statement to modify records in the table
- Use the DELETE statement to remove records from the table

Command #1 – Build a table named *TestScores* with the following columns:

- StudentID Varchar(10) -- text data, up to 10 characters long
- TestCode Varchar(6) -- text data, up to 6 characters long
- TestDate Datetime -- data types for dates vary among RDBMS's
- Score Numeric(9,2) -- number, up to 9 total digits, with 2 of those following the decimal point

```
CREATE TABLE TestScores
(
    StudentID varchar(10)
    , TestCode        varchar(6)
    , TestDate        datetime
    , Score            numeric(9,2)
)
```

Command #2 – Add the following four records to this table:

StudentID	TestCode	TestDate	Score
S00001	ACTC	1-JAN-2010	26
S00002	SATM	3-FEB-2010	640
S00002	SATR	3-FEB-2010	595
S00003	PI	6-MAR-2010	3.14

```
INSERT INTO TestScores
VALUES ('S00001', 'ACTC', '1-JAN-2010', 26);

INSERT INTO TestScores
VALUES ('S00002', 'SATM', '3-FEB-2010', 640);

INSERT INTO TestScores
VALUES ('S00002', 'SATR', '3-FEB-2010', 595);

INSERT INTO TestScores
VALUES ('S00003', 'PI', '6-MAR-2010', 3.14);
```

Command #3 – Perform the following modifications to the records in this table:

- The ACTC test score for student S00001 was originally entered as 26. It should be changed to 28.
- Student S00002 took two components of the SAT on 19-Feb-2010, but it was incorrectly entered as 3-Feb-2010 originally. Change the dates for S00002 for both components to 19-Feb-2010.

```
UPDATE TestScores
  SET Score = 28
  WHERE StudentID = 'S00001';

UPDATE TestScores
  SET TestDate = '19-FEB-2010'
  WHERE StudentID = 'S00002';
```

Command #4 – Delete the 'PI' test score record for Student 'S00003'.

```
DELETE FROM TestScores
  WHERE StudentID = 'S00003'
  AND TestCode = 'PI';
```

